2015年度 第1回 AICS公開ソフト講習会
K MapReduce ハンズオン

滝澤 真一朗
松田 元彦
丸山 直也

理化学研究所 計算科学研究機構
プログラム構成モデル研究チーム

目的

- KMRの導入方法を学ぶ
- KMRRUNを用いたMapReduceプログラム実行方法を学ぶ

ハンズオン環境

- FOCUS Supercomputer
- 計算資源: Aシステム
  - CPU: Intel Xeon L5640 (2.26GHz) x2 (24 Cores in total)
  - Memory: 48 GB
  - Queue名: a024h
  - Staging: 不要
- ストレージ: /home2/gleo
  - 100GB/Group
  - $HOMEではなく、主な作業はここで行います
  - 本日のプログラム・データ置き場

```
/home2/gleo/share/kmr
```

環境設定

- ログイン

```
$ ssh USER_NAME@ssh.j-focus.jp
$ ssh ff
```

<- フロントエンドノードにログイン

- 作業ディレクトリの作成とMPI設定

```
$ mkdir /home2/gleo/`whoami`
$ export WORK=/home2/gleo/`whoami`
$ module load gnu/openmpi165
```

Agenda

- KMRのインストール
- KMRRUNによるMapReduce実行
  - PI計算を例に、逐次プログラム、MPIプログラムをMapper/Reducerとして実行
- KMRRUNによる複数計算の一括実行
  - ゲノム解析プログラムを例に、従来なら複数ジョブとして実行する計算を1ジョブにまとめて実行

Agenda

- KMRのインストール
- KMRRUNによるMapReduce実行
  - PI計算を例に、逐次プログラム、MPIプログラムを Mapper/Reducerとして実行
- KMRRUNによる複数計算の一括実行
  - ゲノム解析プログラムを例に、従来なら複数ジョブとして実行する計算を1ジョブにまとめて実行

KMRのインストール (1/2)

- KMR-1.7 Release (2015-06-22) をインストール
- 展開

```
$ cd $HOME
$ tar zxf ¥(改行しない)
  /home2/gleo/share/kmr/kmr-1.7.tar.gz
$ cd kmr-1.7
```

- configure
  - インストールパスを指定

```
$ ./configure --prefix=$HOME/lib/kmr-1.7
```

- make & install
  - $HOME/lib/kmr-1.7にインストールされる

```
$ make
$ make install
```

KMRのインストール (2/2)

- 環境変数を設定

```
$ export PATH=~/lib/kmr-1.7/bin:$PATH
$ export MANPATH=~/lib/kmr-1.7/man:$MANPATH
```

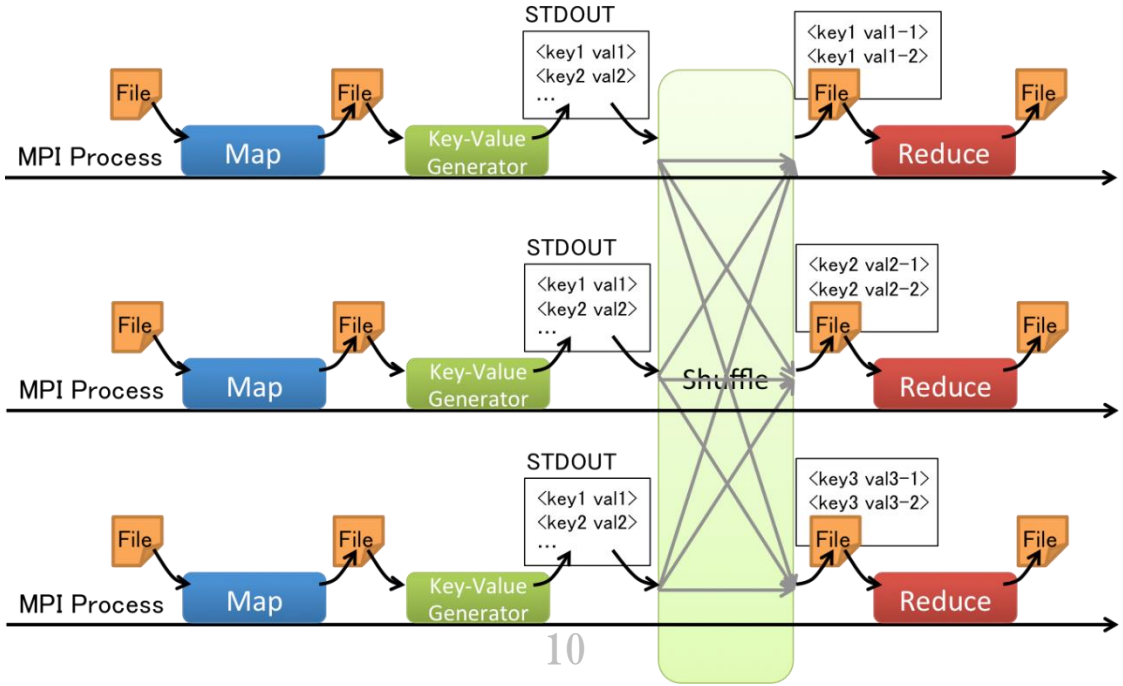
  - シェルの設定ファイル(~/.bashrcなど)に記述しておくと良い

Agenda

- KMRのインストール
- KMRRUNによるMapReduce実行
  - PI計算を例に、逐次プログラム、MPIプログラムをMapper/Reducerとして実行
- KMRRUNによる複数計算の一括実行
  - ゲノム解析プログラムを例に、従来なら複数ジョブとして実行する計算を1ジョブにまとめて実行

KMRRUN

- MapReduceワークフローを実行
- Mapper/Reducerとして、MPIプログラム、任意の言語で実装された逐次プログラム(ノード内並列対応)を実行可能
  - Mapperの出力からKVを生成し、標準出力に書き出す Key-Value Generator プログラムも必要



Mapper/Reducer/KV Generator仕様

	Mapper	KV Generator	Reducer
実装言語	任意	任意	任意
並列実行	MPI/OpenMP	OpenMP	MPI/OpenMP
入力	• ファイル読み込み • ファイル名は最後の引数として渡される	• ファイル読み込み • Mapperの入力ファイル名が最後の引数として渡される • Mapperの出力を読み込む場合は、ファイル名を推測して作成	• ファイル読み込み • ファイル名は最後の引数として渡される • 1 KV/行フォーマット • KeyとValueはスペース1つで区切られる
出力	• ファイル書き出し • ファイル名は、入力ファイル名から類推できる名前とする	• 標準出力に出力 • 1 KV/行フォーマット • KeyとValueはスペース1つで区切られる	• ファイル書き出し

PI計算サンプルプログラム

- PI計算のサンプルプログラムを2種類用意
  - 逐次プログラム版：KMR_SRC/kmrrun/
    - Mapper：pi.mapper.c
    - KV Generator：pi.kvgen.sh
    - Reducer：pi.reducer.c
  - MPIプログラム版：KMR_SRC/kmrrun/
    - Mapper：mpi_pi.mapper.c
    - KV Generator：mpi_pi.kvgen.sh
    - Reducer：mpi_pi.reducer.c
- コンパイル方法

```
$ cd ~/kmr-1.7/kmrrun
$ make pi.mapper pi.reducer
$ make mpi_pi.mapper mpi_pi.reducer
```

PI計算の実装 (1/3)

- Mapper
  - 入力
    - プロットする点の数が書かれた入力ファイル
  - 処理
    - 指定された数分の点をランダムに生成し、半径1の円に入る点の数を数え上げる
  - 出力
    - 「半径1に入る点の数/点の総数」を「入力ファイル名.out」ファイルに書き出す

実行イメージ

```
$ cat ./input/000
10000
$ ./pi.mapper ./input/000
$ ls ./input
000    000.out
$ cat ./input/000.out
7829/10000
```

PI計算の実装 (2/3)

- KV Generator
  - 入力
    - Mapperの入力ファイル
  - 処理
    - Mapperの出力ファイルパスを取得
    - Mapperの出力ファイルを読み込む
    - Mapperの出力ファイルを削除
  - 出力
    - Key-Value Pair <0, ファイルコンテンツ> を標準出力に出力
      - KMRRUN実行時には、KV Generator実行後、自動的にShuffleが行われ、ファイル名「0」ファイルに全てのKVが保存される
      - Keyが1種類なので、Reducerは1ノードで実行される

実行イメージ

```
$ ls ./input
000    000.out
$ ./pi.kvgen.sh ./input/000
0 7829/10000
$ ls ./input
000
```

PI計算の実装 (3/3)

- Reducer
  - 入力
    - 1行に1KVが書かれた入力ファイル
  - 処理
    - 点の数よりPIを計算
  - 出力
    - 計算結果を「pi.out」ファイルに書き出す

実行イメージ

```
$ ls ./
0                    pi.mapper
pi.keygen.sh  pi.reducer
$ cat 0
0 7829/10000
0 7830/10000
$ ./pi.reducer ./0
3.131800
$ ls ./
0                    pi.out
pi.keygen.sh  pi.reducer
pi.mapper
$ cat pi.out
3.131800
```

KMRRUNコマンド

- 実行コマンド

```
$ mpiexec MPIOPT ./kmrrun -n procs -m mapper ¥
-k kvgen -r reducer ./input
```

- コマンドの意味（kmrrunのmanpageも参考）

kmrrun	KMRRUNプログラム本体。 KMR_INST/lib/kmrrunにインストールされている。
−n procs	1回のMapper/Reducer実行で使用するプロセス数を指定。 「m_procs:r_procs」フォーマットで指定すれば、Mapper/Reducerで異なるプロセス数で実行可能。デフォルトは1。 [省略可能]
−m mapper	Mapperプログラム
−k kvgen	KV Generatorプログラム [省略可能]
−r reducer	Reducerプログラム [省略可能]
./input	Mapperの入力ファイル、またはディレクトリ。 全MPIプロセスからアクセスできる、共有ディレクトリ上におくこと。

逐次版PI計算の実行 (1/3)

1. 入力ファイル群を作業ディレクトリに展開

```
$ cd $WORK
$ tar zxf /home2/gleo/share/kmr/150624_kmr_handson.tar.gz
```

2. プログラム群を作業ディレクトリにコピー

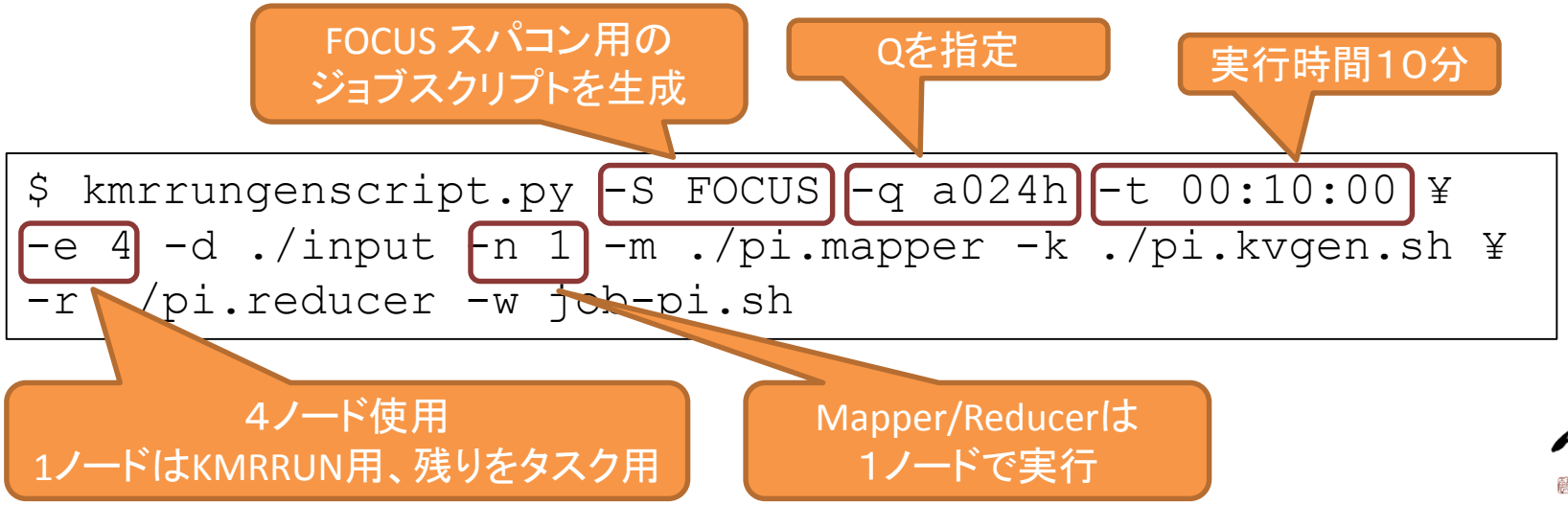
```
$ cd $WORK/150624_kmr_handson/pi
$ cp $HOME/kmr-1.7/kmrrun/pi.mapper .
$ cp $HOME/kmr-1.7/kmrrun/pi.kvgen.sh .
$ cp $HOME/kmr-1.7/kmrrun/pi.reducer .
```

3. ジョブスクリプト作成プログラムを実行

```
$ kmrrungenscript.py -S FOCUS -q a024h -t 00:10:00 ¥
-e 4 -d ./input -n 1 -m ./pi.mapper -k ./pi.kvgen.sh ¥
-r ./pi.reducer -w job-pi.sh
```

KMRRUNヘルパープログラム

- kmrrungenscript.py
  - スケジューラごとのジョブスクリプトを生成
    - 現状、京とFOCUSスパコンをサポート
  - KMRインストールディレクトリ下のbin/に存在
  - 使い方の詳細はコマンドのManpageを参照
- 先の実行例



K computer

逐次版PI計算の実行 (3/3)

5. ジョブ実行

```
$ fjsub job-pi.sh
```

– fjstat コマンドでジョブ実行状況を確認

6. 出力を確認

```
$ ls
job-pi.sh                job-pi.sh.171459.out
job-pi.sh.171459.err     pi.out
```

pi.out	最終出力ファイル
job-pi.171459.out	標準出力
job-pi.171459.err	標準エラー出力

MPI版PI計算の実行

1. プログラムを作業ディレクトリにコピー

```
$ cd $WORK/150624_kmr_handson/pi
$ cp $HOME/kmr-1.7/kmrrun/mpi_pi.mapper .
$ cp $HOME/kmr-1.7/kmrrun/mpi_pi.kvgen.sh .
$ cp $HOME/kmr-1.7/kmrrun/mpi_pi.reducer .
```

- ジョブスクリプト作成プログラムを実行

```
$ kmrrungenscript.py -S FOCUS -q a024h -t 00:10:00 ¥
-e 4 -d ./input -n 2 -m ./mpi_pi.mapper ¥
-k ./mpi_pi.kvgen.sh -r ./mpi_pi.reducer -w job-mpi_pi.sh
```

MPIプログラムでは必ず2以上を指定

- ジョブ実行

```
$ fjsub job-mpi_pi.sh
```

  - 結果は mpi_pi.out に保存

[補足] ノード・プロセス数指定時の注意

- KMRRUNではMPIの動的プロセス生成の仕組みを用いて、Mapper/Reducerを実行しています
- Mapper/Reducerの同時実行数は、生成できる動的プロセス数に依存

  最大動的プロセス数 = 全MPIプロセス数 – 静的プロセス数

  - 全MPIプロセス数（MPI_UNIVERSE_SIZE）
    - MPIプログラム実行時に決定
      - » インタラクティブ実行の場合、Machinefileで指定するノード数
      - » FOCUSスパコンの場合、「#SBATCH -n X」で指定する値
  - 静的プロセス数
    - mpirun、mpiexecの「-n」オプションで指定
  - Mapper/ReducerがMPIプログラムの場合、同時起動できるMapper/Reducer数は並列数に反比例

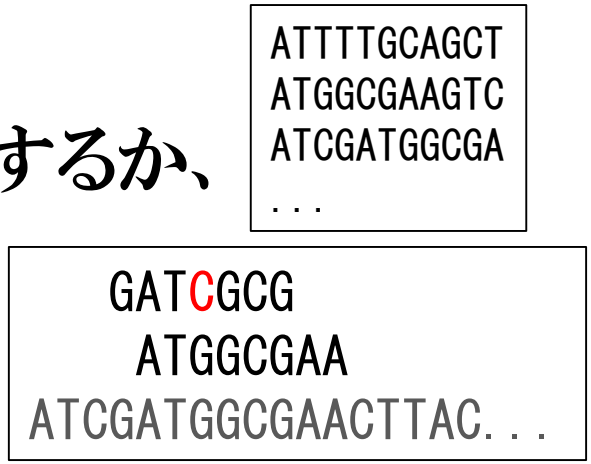
    最大同時起動数 = 最大動的プロセス数 / 1実行の並列数

Agenda

- KMRのインストール
- KMRRUNによるMapReduce実行
  - PI計算を例に、逐次プログラム、MPIプログラムをMapper/Reducerとして実行
- KMRRUNによる複数計算の一括実行
  - ゲノム解析プログラムを例に、従来なら複数ジョブとして実行する計算を1ジョブにまとめて実行

ゲノム解析プログラム

- ゲノムシークエンサーからの大量の出力データを解析
  - 出力は一定長の塩基配列の羅列
  - 既知のゲノム配列のどの部分に該当するか、全配列に対して処理する（アライメント処理）



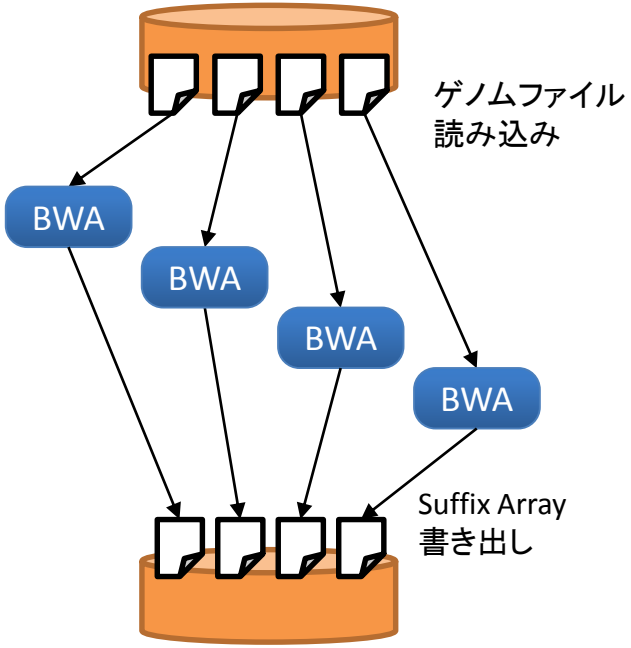
- アライメントツール BWA を、多数に分割されたゲノムデータに対して繰り返し実行
  - ハンズオンでは、入力ゲノムファイルに対してSuffix Arrayを作る処理のみを行う

BWA : http://bio-bwa.sourceforge.net/

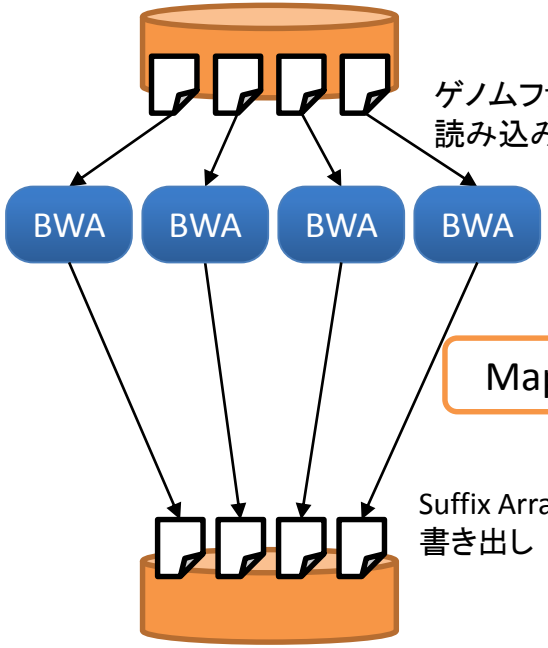
K computer

ゲノム解析プログラム実行イメージ

BWAの実行パターン



KMRRUNへのマッピング



MapとしてBWAを実行

Reduceを行わない
KMRRUNを実行

K computer

BWA実行の実装

- bwa コマンドを実行するラッパープログラムとしてシェルスクリプトで実装
- 入力
  - 参照ゲノム
  - 解析対象ゲノムファイル
- 出力
  - Suffix Arrayファイル
- 処理
  - bwa コマンド(逐次プログラム)のパラメータを設定して実行

実行イメージ

```
$ ls ./
bwa         bwa_task.sh
bwa_db/     input/
$ ls bwa_db
reference.fa.bwt
reference.fa.rbwt
$ ls input
part_1.000    part_1.001
$ ./bwa_task.sh ¥
./bwa_db/reference.fa ¥
./input/part_1.000
...(エラー出力を表示)
$ ls ./
bwa             input/
bwa_db/         output/
bwa_task.sh
$ ls ./output
part_1.000.sai
```

BWAの実行

1. ジョブスクリプト生成プログラムを実行

```
$ cd $WORK/150624_kmr_handson/genome
$ kmrrungenscript.py -S FOCUS -q a024h -t 00:10:00 ¥
-e 5 -d ./input -n 1 ¥
-m "./bwa_task.sh ./bwa_db/reference.fa" -w job-genome.sh
```

2. ジョブ実行

```
$ fjsub job-genome.sh
```

4ファイル並列処理できるように、5プロセス以上を指定（4以下を指定しても動作可能）

クオーテーションでくくり、引数に参照ゲノムファイルを指定

3. 出力を確認

```
$ ls
job-genome.sh.171520.err   job-genome.sh
job-genome.sh.171520.out   output
$ ls output
part_1.000.sai     part_1.002.sai
part_1.001.sai     part_1.003.sai
```

[補足] Python Exampleの実行 (1/2)

- 準備
  - MPI処理系のPython APIをインストール
    - 講習会用に /home2/gleo/share/kmr/python に OpenMPI対応mpi4pyをインストール済み
  - ジョブスクリプトにて、KMRライブラリに環境変数をセット
    - KMR共有ライブラリに対してLD_LIBRARY_PATHをセット
    - kmr4pyモジュールに対してPYTHONPATHをセット
- 実行方法

```
$ cd $HOME/kmr-1.7/ex
$ vi job-kmeans.sh              (next page)
$ fjsub job-kmeans.sh
```

[補足] Python Exampleの実行 (2/2)

```
#!/bin/bash
#
#SBATCH -p a024h
#SBATCH -t 00:20:00
#SBATCH -n 4
#SBATCH -J job-kmeans.sh
#SBATCH -o job-kmeans.sh.%J.out
#SBATCH -e job-kmeans.sh.%J.err

module load gnu/openmpi165
export PYTHONPATH=/home2/gleo/share/kmr/python:$PYTHONPATH
export LD_LIBRARY_PATH=$HOME/lib/kmr-1.7/lib:$LD_LIBRARY_PATH
export PYTHONPATH=$HOME/lib/kmr-1.7/lib:$PYTHONPATH

mpiexec python kmeanspy.py
```

おわり

ご質問・お問い合わせ

丸山 : nmaruyama
松田 : m-matsuda
滝澤 : shinichiro.takizawa @riken.jp

K computer